Efficient substitutes for subdivision surfaces

June 1, 2009

Abstract

This course provides an introduction to Approximate Subdivision Surfaces, an overview of the most recent theoretical results and their implementations on the current and next-generation GPUs, and a demonstration of these techniques and their applications in the game and movie industries.

1 Fundamentals of efficient substitutes for Catmull-Clark subdivision surfaces

(Jörg Peters)

As real time graphics aspire to movie-quality rendering, *higher-order, smooth surface representations* take center stage. Besides tensor-product splines, Catmull-Clark subdivision has become a widely accepted standard – whose advantages we now want to replicate in real-time environments.

Recently, efficient substitutes for recursive subdivision have been embraced by the industry. These notes discuss the theory justifying the use of efficient substitutes for recursive subdivision. (Three other sections discuss their current and future support in the graphics pipeline, in the movie production pipeline and for gaming implementations.)

Below we therefore explore the motivation and the properties that surfaces and representations should satisfy to be used alongside or in place of Catmull-Clark subdivision.

1.1 Why do we want smooth surfaces?

The ability to have continuously changing normals (complemented by creases where we choose to have them) is important both artistically and to avoid errors in downstream algorithms.

Artistic shape considerations require the ability to create smooth surfaces and transitions: sharp turns and sharply changing normals do not match our experience of, say, faces and limbs. On the other hand, where the curvature is high compared to the surroundings, smoothed out *creases* are often crucial to bring to life an object or an animation character.

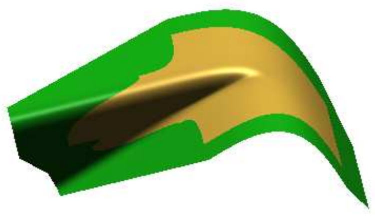
Figure 1: **Smoothness and creases** (from [PK09]).

Downstream algorithms convey realism via lighting, silhouettes, and various forms of texturing. In particular, the diffuse and specular components of the lighting computation rely on well-defined directions (normals) $\mathbf{n}$ associated with points $\mathbf{v}$ of the object. This is evident in the OpenGL lighting model. [1] Downstream algorithms relying on $\mathbf{n}$ and $\mathbf{v}$ include (click on the hyperrefs if you have the electronic version of the notes) are for example

– Gouraud shading (http://en.wikipedia.org/wiki/Gouraud_shading)

– Phong shading (http://en.wikipedia.org/wiki/Phong_shading)

– Bump mapping(http://en.wikipedia.org/wiki/Bump_mapping)

– higher resultion near the silhouette (http://en.wikipedia.org/wiki/Silhouette)

– Normal mapping (http://en.wikipedia.org/wiki/Normal_mapping)

– Displacement mapping (http://en.wikipedia.org/wiki/Displacement_mapping)

[1] The red, green or blue intensity of OpenGL lighting is

$$\begin{aligned}\text{intensity } := &\text{emission}_m + \text{ambient}_l \cdot \text{ambient}_m \\ &+ \sum_{\text{lights}} \frac{1}{k_0 + k_1 d + k_2 d^2} \cdot \text{spot}_\ell \cdot \Big(\text{ambient}_\ell \cdot \text{ambient}_m \dots \\ &\dots + \max\{(\mathbf{p}-\mathbf{v})\cdot\mathbf{n}, 0\} \cdot \text{diffuse}_\ell \cdot \text{diffuse}_m + \quad \max\{\mathbf{s}\cdot\mathbf{n}, 0\}^{\text{shininess}} \cdot \text{specular}_\ell \cdot \text{specular}_m\Big)\end{aligned}$$

	$\mathbf{v}$	vertex	$\mathbf{n}$	normal	$\mathbf{p}$	light position	$\mathbf{e}$	eye position
where	m	material	ℓ	light source	l	lighting model	d	$\Vert\mathbf{p}-\mathbf{v}\Vert$
	$\mathbf{s} := \frac{\mathbf{s}'}{\Vert\mathbf{s}'\Vert}$	$\mathbf{s}' := \frac{\mathbf{v}-\mathbf{p}}{\Vert\mathbf{v}-\mathbf{p}\Vert} + \frac{\mathbf{v}-\mathbf{e}}{\Vert\mathbf{v}-\mathbf{e}\Vert}$						

1.2 Surface smoothness

Surfaces that can locally be parameterized over their tangent plane are called regular C^1 surfaces (or manifolds). Such surfaces provide a unique normal $\mathbf{n}$ at every point computable as the cross product of two independent directions $\mathbf{t}_1$ and $\mathbf{t}_2$ in the tangent plane: $\mathbf{n}||\mathbf{t}_1 \times \mathbf{t}_2$. To characterize smoothness of piecewise surfaces, as they occur in graphics applications, the area of geometric modeling (http://www.siam.org/activity/gd) has developed the notion of 'geometric continuity'. Essentially, two patches $\mathbf{a}$ and $\mathbf{b}$ join G^1 to form a part of a C^1 surface if their (partial) derivatives match along a common curve *after a change of variables*.

Formally (see e.g. [Pet02]), the patches $\mathbf{a}$ and $\mathbf{b}$ map a subset $\square$ of $\mathbb{R}^2$ to $\mathbb{R}^3$. That is $\mathbf{a}, \mathbf{b} : \square \subsetneq \mathbb{R}^2 \to \mathbb{R}^3$. Let ρ be a map that suitably connects the domains, i.e. changes the variables. We call such a ρ a (regular) reparameterization $\rho : \mathbb{R}^2 \to \mathbb{R}^2$. Let $E = [0..1] \times 0$ be an edge of $\square$ and $\mathbb{Z}$ the non-negative integers and let $\circ$ denote composition, i.e. the image of the function on its right provides the parameters of the function to its left.
Patches $\mathbf{a}$ and $\mathbf{b}$ join G^k if there exists a (regular) reparameterization ρ so that for the parameter restricted to E

$$\text{for } i, j \in \mathbb{Z}, i + j \le k, \qquad \partial_1^i \partial_2^j\, \mathbf{a} \circ \rho = \partial_1^i \partial_2^j\, \mathbf{b}. \tag{1}$$

Smooth surfaces must in particular satisfy G^1 continuity, i.e. (1) for $k = 1$. That is the surfaces need continuity along the common curve and matching transversal derivatives (across the edge):

$$\mathbf{a} \circ \rho(E) = \mathbf{b}(E), \quad \partial_2^j\, \mathbf{a} \circ \rho(E) = \partial_2^j\, \mathbf{b}(E). \tag{2}$$

(Matching derivatives along the boundary curve, $\partial_1^j\, \mathbf{a} \circ \rho(E) = \partial_1^j\, \mathbf{b}(E)$ already follow from $\mathbf{a} \circ \rho(E) = \mathbf{b}(E)$.) Proofs are therefore usually concerned with establishing $\partial_2^j\, \mathbf{a} \circ \rho(E) = \partial_2^j\, \mathbf{b}(E)$ for patches with a common boundary curve (segment).) When $\mathbf{a}$ and $\mathbf{b}$ are polynomial patches, (2) amounts to enforcing linear equations on the coefficients *when ρ has been selected*.

To join n patches G^k at a vertex, two additional constraints come into play: (a) the *vertex enclosure constraint* must hold for the normal component of the boundary curves; and (b) the reparameterizations ρ must satisfy a *consistency constraint*. Both constraints arise from the periodicity when visiting the patches, respectively the reparameterizations surrounding a vertex. For a detailed explanation of these constraints and an in-depth look at geometric continuity see for example [Pet02].

A complex of G^1-connected patches admits a C^1 manifold structure. G^1 constructions differ from the approach of classical differential geometry in that they do not require fully defined charts. G^1 continuity only regulates differential quantities along an interface, whereas charts require overlapping domains.

1.3 Filling the normal channel

The separation of the position and the normal channel in the graphics pipeline makes it possible to substitute for the true normal field of the surface, a field not necessarily orthogonal to the surface. This 'field of directions' can be used, as in bump mapping, to make a surface less smooth or to make it appear smoother (under lighting but not its silhouette) than it truly is.

Of course, the geometry and the shape implied by lighting with the 'field of directions' declared to be the 'normal field' will be (slightly) inconsistent. But we may hope that this does not attract attention (see PN triangles and

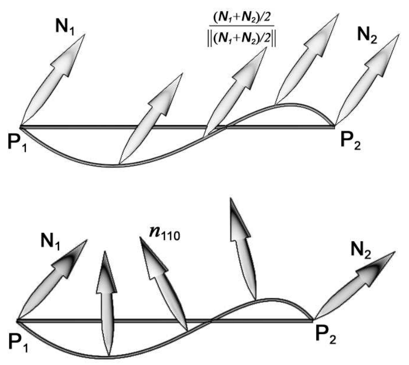

Figure 2: **Normal channel** defined separate from the geometry (from [VPBM01]). Linear interpolation of the normals at the endpoints (*top*) ignores inflections in the curve while the quadratic normal construction (*bottom*) can pick up such shape variations.

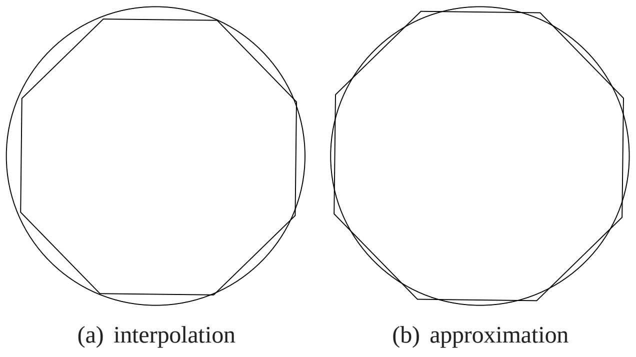

Figure 3: Which polygon represents the circle better?

ACC patches below). The visual impact of the 'field of directions' in the normal channel may be so good that a careful designer will have to check surface quality partly by examining silhouettes.

For polyhedral models, determining vertex normals is an underconstrained problem and various heuristics, such as averaging face normals, can be used to fill the normal channel (for a comparison see e.g. [JLW05]). Figure 2 illustrates that some level of consistency of the normal channel with the true surface geometry is important. Substitutes of subdivision surfaces (Section 1.8) therefore typically use more sophisticated approaches to fill the normal channel.

1.4 Evaluation or approximation?

Due to the pixel resolution, we ultimately render an averaged, linearized *approximation* of surfaces. As Figure 3 illustrates, exact evaluation followed by piecewise linear completion need not be superior to any other approximation where no point lies exactly on the circle. For another example in 2D consider the U-shape $y := x^2$ for $x \in [-1..1]$. The line segment that connects $(-1, y(-1))$ to $(1, y(1))$ is based on exact evaluation at the parameters -1 and 1 but is a much poorer approximation (in the max-norm) to the parabola piece than the line segment $(-1, 1/2)$ to $(1, 1/2)$.

On a philosophical level, if one ultimately renders a triangulation of the surface, there is no reason to believe that a triangulation with exact values at the vertices is a 'best' approximation to the true surface. All we know is that the maximal error does not occur at the vertices but in the interior of the approximating triangles. The error in the interior of the triangle may be far more than the distance between a control point and the surface or a control triangle and the surface.

So, while 'exact' evaluation may sound better than 'approximate' evaluation, there is often no reason to prefer one to the other. In fact, if we stay with the control net of a surface rather than projecting it to the limit, we preserve the full information of the spline or subdivision representation.

One attempt at quantifying and minimizing this error are *mid-structures* of Subdividable Linear Efficient Function Enclosures (slefes) [Pet04]. Mid-structures link the curved geometry of the surface to a two-sided (sandwiching) piecewise linear approximation. For a subclass of surfaces the approximation is optimal in the max-norm (http://en.wikipedia.org/wiki/Supremum_norm).

The main justification for positioning points as exactly as possible on a surface is that, when two abutting patches are tessellated independently, it is good to agree on a rule that yields the same point in $\mathbb{R}^3$ so that the resulting surface has no holes, i.e. is *watertight*. Mandating the point to be exactly on the surface (and being careful in its computation) is an easy-to-agree-on strategy for a consistent set of points. Of course, any other well-known rule of approximation would do as well.

1.5 Polynomial patches of degree bi-3 and bi-cubic splines

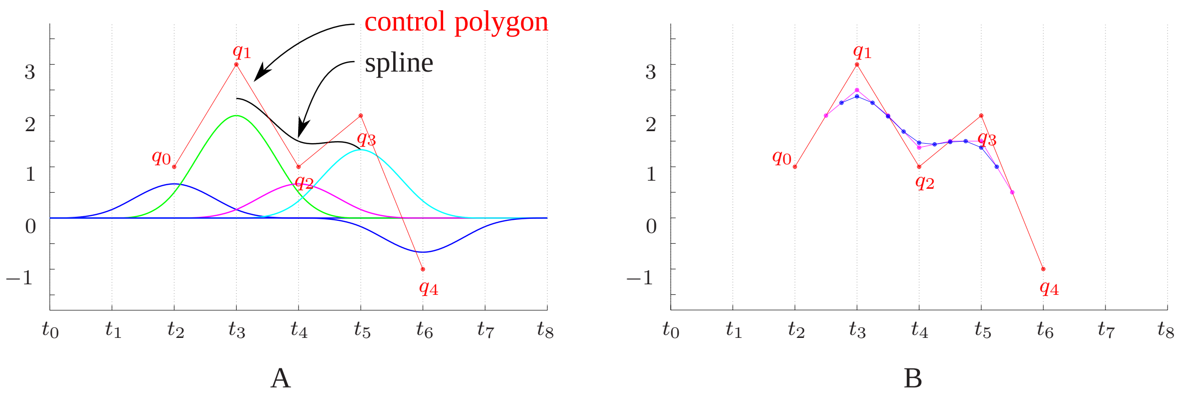

Figure 4: **Univariate uniform cubic spline** (from [Myl08]). (A) Control points $q := [1, 3, 1, 2, -1]$ (*red*) and knots $\mathbf{t} := [-1, 0, 1, 2, 3, 4, 5, 6, 7]$ define a cubic spline $\mathbf{x}(t)$ as the sum of uniform B-spline bases f_ℓ scaled by their respective control points (*blue, green, magenta, cyan*). (B) An equivalent definition of the spline is as the limit of iterative control polygon refinement (subdivision).

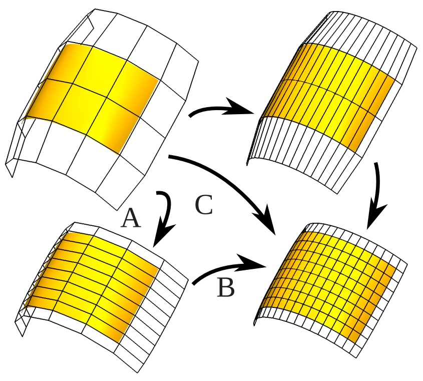

Figure 5: Commutativity of **tensor-product spline subdivision** (from [MKP07]). Bi-3 spline subdivision (A) in one direction followed by (B) the other, or (C) simultaneous refinement as in Catmull-Clark.

If we want to avoid linearization, we need to use quadratic patches at a minimum. Quadratics offer a rich source of shapes – after all C^2 surfaces can locally be well-approximated by them) but smoothly stitching pieces together is generally only possible for regular partitions. Moreover, enforcing G^1 continuity can force flat spots for higher-order saddles, such as a monkey saddle (http://en.wikipedia.org/wiki/Monkey_saddle). [PR98] lists all classes of quadratic shapes.

Many curved objects are therefore modeled with cubic splines $\mathbf{x}(t) := \sum_\ell q_\ell f_\ell(t)$ as illustrated in Figure 4. Cubic spline curves in B-spline form are available in OpenGL as `gluNurbsCurve`. By tracing out cubic splines in two independent variables (u, v), we obtain a tensor-product spline available in OpenGL as `gluNurbsSurface`. We call the tensor of cubic splines *bi-3 spline* or bi-cubic spline:

$$\sum_{i=0}^{3}\sum_{j=0}^{3} q_{i,j} f_i(u) f_j(v). \tag{3}$$

Bi-3 splines in *B-spline form* can be evaluated efficiently, for example by de Boor's algorithm (http://en.wikipedia.org/wiki/De_Boor_algorithm).

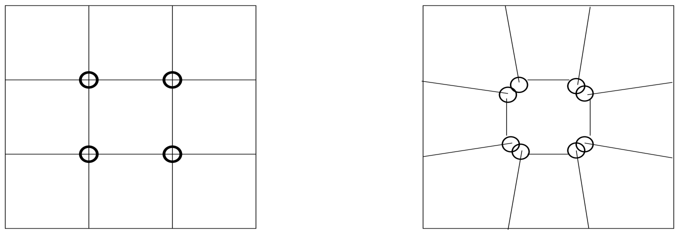

Figure 7: Control point structure of (*left*) a polynomial bi-3 patch and (*right*) a **Gregory patch**.

Just as for curves, each tensor-product spline can be split into its parts by averaging control points. This is the C^2 bicubic subdivision as illustrated in Figure 5 whose limit is the bi-3 spline patch.

An alternative representation of a polynomial piece is the *Bernstein-Bezier form* or, shorter, BB-form [2]. Cubic spline curves in B-spline form are available in OpenGL as `glMap1`. It, too, can be tensored

$$\sum_{i=0}^{3}\sum_{j=0}^{3} b_{i,j}h_i(u)h_j(v), \quad h_k(t) := \frac{3!}{(3-k)!k!}(1-t)^{3-k}t^k. \tag{4}$$

Bi-3 splines in BB-form are available in OpenGL as `glMap2`. Every surface in B-spline form can be represented in BB-form using one patch in BB-form for every quadrilateral of the B-spline control net. Due to combinatorial symmetry in the positions, there are three types of formulas in B-form to BB-form conversion:

$$\begin{aligned} 9b_{11} &:= 4q_{11} + 2(q_{12} + 2q_{21}) + q_{22}, \qquad (5)\\ 12b_{10} &:= 4(q_{11} + q_{21}) + q_{10} + q_{20} + q_{12} + q_{22}, \\ 36b_{00} &:= 16q_{11} + 4(q_{21} + q_{12} + q_{01} + q_{10}) + q_{22} + q_{02} + q_{00} + q_{20}. \end{aligned}$$

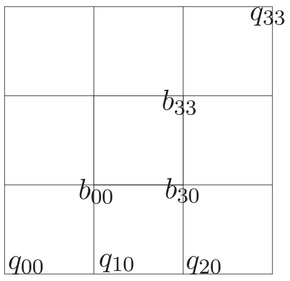

Figure 6: Bi-3 **conversion** from B-spline coefficients q_{ij} to BB-coefficients b_{ij}

Conversely, if patches in BB-form are arranged in checkerboard form, they can be represented in B-spline form. To obtain the simplest representation, we remove knots where the surface is sufficiently smooth. (If we do this locally and carefully keep track of where we removed knots, we arrive at T-splines [SZBN03]). That is, the B-spline form and the BB-form are equally powerful, but one may choose B-splines to have fewer coefficients and built-in smoothness, while the BB-form provides interpolation at the corners.

Additionally, the BB-form can be generalized to two variables so that the natural domain is a triangle, i.e. to total degree BB-form [3]. Polynomials in BB-form can be evaluated by de Casteljau's algorithm (http://en.wikipedia.org/wiki/D
As a byproduct of evaluation, De Casteljau's algorithm provides the derivatives at the evaluation point from which the normal direction can be obtained by a simple cross product. For a detailed exposition of these useful representations see the textbooks [Far97, PBP02].

There are a number of classic bi-3 surface constructions [Bez77, vW86, Pet91], but, due to fundamental lower bounds, they work in general only if we split facets into several polynomial pieces.

The C^1 bi-3 *Gregory patch* [Gre74, BG75] is a rational surface patch $\mathbf{x} : [0..1]^2 \to \mathbb{R}^3$ such that $\partial_u\partial_v\mathbf{x} \neq \partial_v\partial_u\mathbf{x}$ can hold at the corners. This allows separate definition of first order derivatives along the two edges emanating from a corner point; this can be viewed as splitting certain control points into two (see Figure 7). The resulting lack of higher-order smoothness contributed to it not being widely used in geometric design but should not be a problem for real time graphics. High evaluation cost and cost of computing normals require careful use.

[2]http://en.wikipedia.org/wiki/Bezier_curve
[3]http://wapedia.mobi/en/Bezier_triangle

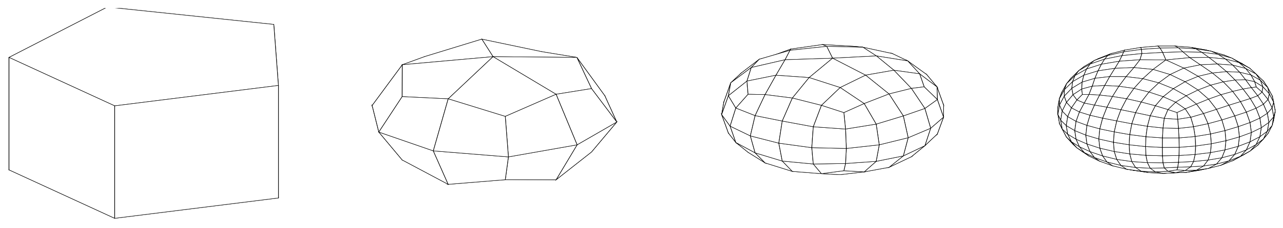

Figure 8: Mesh refinement by the **Catmull-Clark** algorithm.

1.6 Subdivision surfaces

We sketch here only the basics of subdivision surfaces [4] sufficient to explain their evaluation and approximation. A full account of the mathematical structure of subdivision surfaces can be found in [PR08]. The SIGGRAPH course notes [ZS00] of Schröder and Zorin, and the book 'Subdivision Methods for Geometric design' by Warren and Weimer [WW02] complement the more formal analysis by a collection of applications and data structures. See also the generic CGAL implementation [SAUK04].

Algorithmically, subdivision presents itself as a mesh refinement procedure that applies rules to determine (a) the position of new mesh points from old ones and (b) the new connectivity. These rules are often represented graphically as weights (summing to one) associated with a local graph or *stencil* that links the old mesh points combined to form one new one: Forming the weighted old mesh points yields the new point. On the GPU, recursive subdivision naturally maps to several shader passes (see e.g. [SJP05, Bun05][5]).

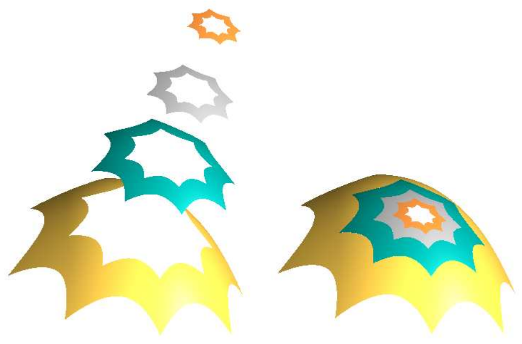

Figure 9: **Subdivision surfaces** consist of a nested sequence of surface rings.

Alternatively, the weights can be arranged as a row of a *subdivision matrix* A. This subdivision matrix maps a mesh of initial points $\mathbf{q}_\ell \in \mathbb{R}^3$ collected into a vector $\mathbf{q}$ to an (m times) refined mesh

$$\mathbf{q}^m = A^m \mathbf{q}. \tag{6}$$

The mesh can have *extraordinary points*. An extraordinary point is one that has an unusual number of direct neighbors n; n is often referred to as the *valence* of the extraordinary point. For example, $n \neq 4$ is unusual for Catmull-Clark subdivision (see Figure 8).

Mathematically, however, a subdivision surface is a spline surface with isolated singularities. Each singularity is the limit of one extraordinary point under subdivision. In particular, the neighborhood of any such singularity consists of a nested sequence of surface rings as illustrated in Figure 9.

In the case of Catmull-Clark subdivision, the nested surface rings consist of n L-shaped *sectors* with three bi-3 polynomial pieces each. Let $\square := [0..1]^2$ be the unit square. Then each sector of the mth ring can be associated with a parameter range $\frac{1}{2^m}(\square - \frac{1}{2}\square)$ (see Figures 4.2, 4.3, 4.4, 4.5 of [PR08] for a nice illustration of this natural parameterization and the fact that the union of rings then forms a spline with a central singularity). An alternative parameterization associates $\lambda_n^m(\square - \lambda_n \square)$ with an L-segment, where λ_n is the subdominant eigenvalue of A for valence n.

[4] subdivision surfaces (http://en.wikipedia.org/wiki/Subdivision_surface)
[5] http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter07.html

1.7 Evaluation of subdivision surfaces

Since subdivision surfaces are splines with singularities there are a number of evaluation methods that also work near extraordinary points. We list four methods below.

1.7.1 Standard Evaluation

(i) determine the ring m (by taking the logarithm base 2);
(ii) apply m subdivision steps (either by matrix or stencil applications);
(iii) interpret the resulting control net at level m as those of the n L-shaped sectors in B-spline form; and
(iv) evaluate the bi-3 spline (by de Boor's algorithm).

While step (ii) seems to require recursion, it can be replaced by the (*non-recursive*) matrix multiplication (6).

This is typically the *most efficient strategy to evaluate a subdivision surface* (and it can not be patented ;-)). It is particularly efficient when many points on a regular grid are to be evaluated, for example when, for even coverage, we want to evaluate 4 times more points in ring m than in ring $m-1$. It is also most efficient when the surfaces have adjustable creases [DKT98], i.e. where Catmull-Clark refinement rules are averaged with curve refinement rules.

Some special scenarios, however, invite different evaluation strategies. Before settling for a strategy, it is good to verify the conditions under which they are appropriate and efficient.

1.7.2 Tabulation of Generating Functions

If the crease ratios are restricted to a few cases *and* the depth of the subdivision is restricted, then we can trade storage for speed by pre-tabulating the evaluation. The idea is to write the subdivision surface $\mathbf{x}$ locally, in the neighborhood of an extraordinary point, as

$$\mathbf{x}(u,v,j) = \sum_{\ell}^{L} \mathbf{q}_\ell b_\ell(u,v,j), \tag{7}$$

where the $\mathbf{q}_\ell \in \mathbb{R}^3$ are the subdivision input mesh points; each $b_\ell \in \mathbb{R}$ is a generating spline, i.e. a function that we may think of as obtained by applying the rules of subdivision considering one coordinate q_j of $\mathbf{q}_j$ and setting all $q_j = 0$ except for q_ℓ; and the summation by ℓ is over all b_ℓ that are nonzero at the point (u,v,j) of evaluation; $j \in \{1,2,\ldots,n\}$ denotes one of the n sectors of the spline (ring). If, for each valence separately, we pre-tabulate the $b_\ell(u,v,j)$ for $\ell = 1,\ldots,L$ then we can look up and combine these values with the subdivision input mesh points $\mathbf{q}_\ell$ at run-time. When stored as textures, approximate 'in-between' values can be obtained by bi-linear averaging. [BS02]

1.7.3 Patch selection (ii) in eigenspace

If several but irregularly distributed parameters are to be evaluated *and* if they lie very close to the extraordinary point, it is worth converting the subdivision input mesh points $\mathbf{q}_\ell$ to eigencoefficients $\mathbf{p}_\ell \in \mathbb{R}^3$. For this, we need to form the Jordan decomposition $A^m = VJ^mV^{-1}$ (just once for any given subdivision matrix A of valence n) and set $\mathbf{p} := V^{-1}\mathbf{q}$ so that

$$A^m\mathbf{q} = VJ^m\mathbf{p}. \tag{8}$$

If the Jordan matrix J^m is diagonal then the computational effort at run time of step (ii) reduces to taking mth powers of its diagonal entries [DS78]. In step (iii) we need to apply V to $\mathbf{p}$ and can then proceed as before with step (iv) to evaluate a bi-3 spline [Sta98]. Note that this method is no more exact than any of the other evaluation methods and that exact evaluation at individual points does not mean that a polyhedron based on the values exactly matches the non-linear subdivision limit surface.

1.7.4 Eigensystem evaluation

For parameters on a grid, Cavaretta et al. showed that, for functions satisfying refinement relations, the exact values on a lattice can be computed by solving an eigenvalue problem [CDM91, page 18],[de 93, page 11]. Schaefer and Warren [SW07] apply this approach to irregular settings.

We note that neither the standard evaluation using (6) nor any of the three approaches just listed require recursion or uniform refinement (with its concomitant high use of memory and possibly of CPU–GPU bandwidth). However, they do not provide convenient short formulas.

1.8 Can it be done simpler? Efficient Substitutes

A surface construction can provide a substitute for the subdivision algorithm if the resulting surfaces have similar properties.

1.8.1 Control polyhedra and proxy splines

The classic substitute is to render, at a finite level of resolution, either the refined control polyhedron or a polyhedron obtained by projecting the refined control vertices to the limit (using the left eigenvectors of the subdivision matrix A). This is based on the fact that the distance between control polyhedron and limit surfaces decreases fast. One of the challenges here is to correctly estimate the distance of the (projected) control polyhedron to the surface in order to determine the (adaptive) subdivision level that gives sufficient resolution for the application. By characterizing control polyhedra as (the images of) proxy splines with the same structure as subdivision surfaces, [PR08, Chapter 8] gives general bounds on this distance for all subdivision schemes. Tighter bounds, specifically for Catmull-Clark subdivision surfaces can be found in [PW08]. Also available is a plug-in by Wu for (pov-)ray tracing based on the bounds in [WP04, WP05]. This class of substitutes is only efficient, if it can be applied adaptively (see, e.g. [Bun05]).

1.8.2 Separate geometry and normal channels

A second class of substitutes takes advantage of the separation of the position and the normal channel in the graphics pipeline. That is, the entries in the normal channel are only approximately ‘normal’ to the (geometry of the) surface.

- **original geometry, refined normals** To create a denser field for the normal channel then would be used by Gouraud shading, we can apply subdivision (averaging) to the polyhedral normals [AB08].
- **refined geometry, refined normals** Replacing an input triangle with normals specified at its vertices, *PN triangles* [VPBM01] consist of a total degree 3 geometry patch that joins continuously with its neighbor and has a common normal at the vertices. To convey the impression of smoothness, a separate quadratic normal patch interpolates the vertex normals (Figure 10). By reducing the patch degree to quadratics, trades flexibility of the geometry for faster evaluation [BA08] (see also [BS07]). Since the quadratic pieces have no inflections

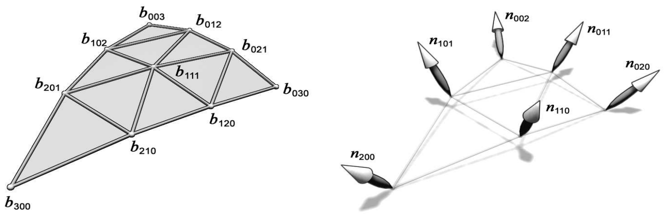

Figure 10: Control point structure of **PN triangles** (from [VPBM01]). (*left*) the positional channel; (*right*) the normal channel.

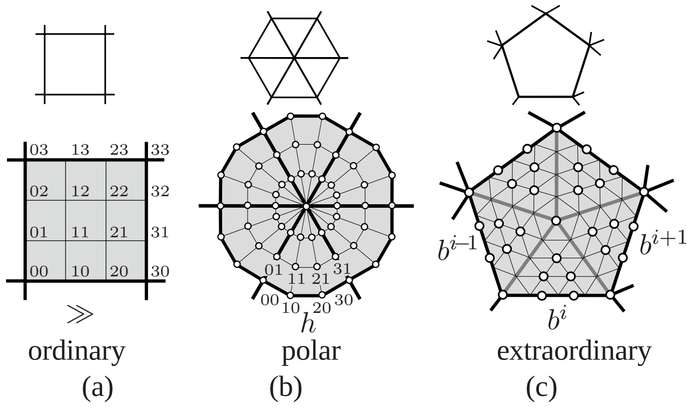

Figure 11: **Mesh-to-patch conversion.** (from [MNP08]) The input mesh (*top*) is converted to patches (*bottom*) as follows. (a) An ordinary facet is converted to a bi-cubic patch with 16 control points f_{ij}. (b) Every triangle in polar configuration becomes a singular bi-cubic patch represented by 13 control points $\circ$. (c) An extra-ordinary facet with n sides is converted to a P_n-patch defined by $6n+1$ control points shown as $\circ$. The P_n-patch is equivalent to n C^1-connected degree-4 triangular patches b^i, $i = 0 \dots n–1$, having cubic outer boundaries.

this is particularly useful when the triangulation is already more refined.

For four-sided facets, the corresponding (family of) *PN quads* was known but not published at the time of PN triangles. Just like the triangles, its bi-3 patches are constructed based solely on the points $\mathbf{v}$ and normals $\mathbf{n}$ at the patch vertices so that a patch need not look up the neighbor quads.

Better shape can be achieved, when the neighbor patch(es) can be accessed. For example, the inner BB coefficients b_{ij} can be derived from a bi-3 spline [Pet08]. One can use Equations 5 for the inner coefficients of type b_{11} and set b_{10} on an edge between two patches as an average of their closest inner points. A good heuristic is to set the corner control points to the Catmull-Clark limit point (with q_0 the central control point and for $\ell = 0, \dots, n-1$ $q_{2\ell-1}$ the direct neighbor points and $q_{2\ell}$ the face neighbor points):

$$n(n+5)b_{00}^{CC} := \sum_{l=0}^{n-1} (nq_{00} + 4q_{2\ell-1} + q_{2\ell})\,. \tag{9}$$

Up to perturbation of interior control points near extraordinary points,

$$(n+5)\, b_{11}^{ACC} := nq_{11} + 2(q_{12} + 2q_{21}) + q_{22}, \tag{10}$$

this is how ACC patches [LS08a] are derived (see also the Section 2.3 of these lecture notes).

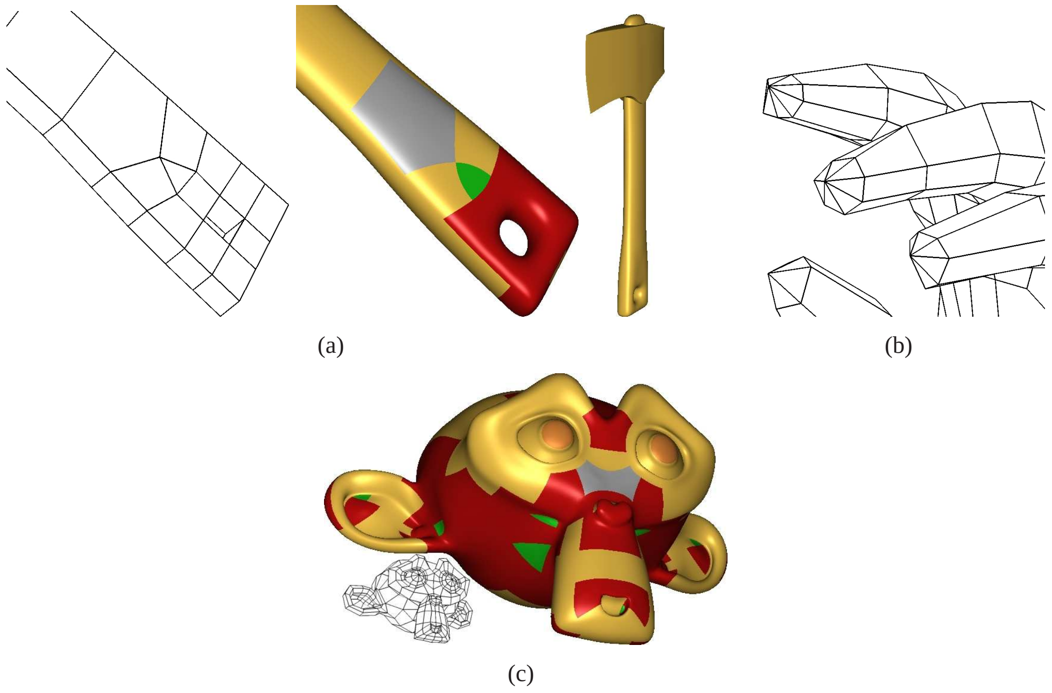

Figure 12: **Quad/tri/pent polar models** (from [MNP08]) (a) Axe handle; using a triangle and a pentagon to transition between detailed and coarser areas. The axe head (*left*) features a sharp crease. (b) Polar configurations naturally terminate parallel feature lines along elongations, like fingers. (c) smooth surface consisting of bi-cubic patches (*yellow*), polar patches (*orange*), and p-patches with $n = 3$ (*green*), $n = 4$ (*red*), $n = 5$ (*gray*).

1.8.3 C^1 surface constructions

A third class of substitutes are proper C^1 surfaces, i.e. their normals can be computed everywhere (eg in the pixel shader) as the cross product of tangents (derivatives obtained as a byproduct of de Casteljau's evaluation) without recourse to a separate normal channel.

These patches are typically polynomial, although a rational construction like Gregory's patch and its triangular equivalent could be used just as well. The patch corners and normals can moreover be adjusted to approximate Catmull-Clark limit surfaces.

Just as the second class, c-patches [YNM$^+$] and the $m$any-sided p(m)-patches [MNP08] (Figures 11 and 12) can be constructed and displayed in real time. [MNP08] comes with shader code, allows for (rounded) creases and polar configurations (see Figure 12(d)) [6] The third class of surface constructions is related to surface splines [Pet95] and Loop's construction [Loo92] and localized hierarchical surface splines [GP99].

1.9 Efficiency

Whether a particular representation or evaluation strategy is time and space efficient depends on the software/hardware setup. However, we can observe the following in the context of GPU rendering.

Fixed, fine triangulations are expensive to transfer to the GPU and require animation of each vertex. They lack refinability. Subdivision surfaces approximated by *recursive refinement*, possibly followed by projection of the control

[6] One concern is that such creases and polar configurations result in 'parametric distortion' when texture mapping. Applying the same crease or polar mapping (in $\mathbb{R}^2$) when looking up texture coordinates, however, shows this concern to be unfounded.

points to their limit, require multiple passes with increasing bandwidth and intermediate memory storage. Subdivision surfaces approximated by *non-recursive evaluation* as listed in Section 1.7 requires the inversion of (moderately sized) matrices. These matrices need to be adapted for different types of creases. Subdivision surfaces approximated by *tabulation* require storage that limits the representable crease configurations. The (efficient) substitutes listed in Section 1.8 allow for creases, adaptive evaluation (by instancing or the tessellation engine) and, as low degree polynomials, have been created to be both space-efficient and time-efficient, in their construction as well as in their evaluation.

efficiency	space	time	comment
triangulation	–	–	fixed resolution
recursive subD	–		adaptivity?
non-recursive subD		–	creases?
tabulation	–	+	creases?
efficient substitutes	+	+	crease✓, adapt✓

1.10 Higher-quality surfaces?

For high-end design, C^1 continuity is not sufficient. One can feel (and sometimes see) the lack of curvature continuity. In fact, Catmull-Clark subdivision does not meet the requirements of high-end design: Generically, near extraordinary points, the curvature lines diverge, and the surfaces becomes hyperbolic [KPR04]. Guided surfacing [KP07, KPN1], Loop and Schaefer [Loo04, LS08b] and most recently a bi-3 C^2 polar subdivision [MP09] promise better shape. Yet, it is not clear that real-time or movie applications can benefit from such high-quality surfaces.

Curiously, at least formally, displacement mapping, which often increase roughness of the surfaces, formally requires derivatives of normals and therefore higher-order continuity.

1.11 Summary

Besides the classical rendering of the control polyhedron, possibly projected onto the surface, there are two classes of surface constructions that can be used as efficient substitutes of subdivision surfaces or as primitives in their own right. Both triangular patches and quad patches are available (as well as polar configurations) to give the designer broad-ranging options and mimic both Catmull-Clark and triangle-based subdivision. The next chapters will explain the use of these constructions in more detail and may inspire additional short-cuts and innovations (see for example [7]), made all the more relevant by the imminent availability of tessellation hardware.

[7]http://castano.ludicon.com/blog/2009/01/07/approximate-subdivision-shading/

References

[AB08] Marc Alexa and Tamy Boubekeur. Subdivision shading. *ACM Trans. Graph, Siggraph Asia*, 27(5):142, 2008.

[BA08] Tamy Boubekeur and Marc Alexa. Phong tessellation. *ACM Trans. Graph, Siggraph Asia*, 27(5):141, 2008.

[Bez77] Pierre E. Bezier. *Essai de definition numerique des courbes et des surfaces experimentales*. Ph.d. thesis, Universite Pierre et Marie Curie, February 1977.

[BG75] R. Barnhill and J. Gregory. Compatible smooth interpolation in triangles. *J of Approx. Theory*, 15(3):214–225, 1975.

[BS02] J. Bolz and P. Schröder. Rapid evaluation of Catmull-Clark subdivision surfaces. In *Proceedings of the Web3D 2002 Symposium*, pages 11–18. ACM Press, 2002.

[BS07] Tamy Boubekeur and Christophe Schlick. QAS: Real-time quadratic approximation of subdivision surfaces. In Marc Alexa, Steven J. Gortler, and Tao Ju, editors, *Proceedings of the Pacific Conference on Computer Graphics and Applications, Pacific Graphics 2007, Maui, Hawaii, USA, October 29 - November 2, 2007*, pages 453–456. IEEE Computer Society, 2007.

[Bun05] Michael Bunnell. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, chapter Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping. Addison-Wesley, Reading, MA, 2005.

[CDM91] A.S. Cavaretta, W. Dahmen, and C.A. Micchelli. Stationary subdivision. *Memoirs of the American Mathematical Society*, 93(453):1–186, 1991.

[de 93] Carl de Boor. On the evaluation of box splines. *Numerical Algorithms*, 5(1–4):5–23, 1993.

[DKT98] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 1998. ACM Press.

[DS78] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10:356–360, September 1978.

[Far97] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. Academic Press, pub-ACADEMIC:adr, fourth edition, 1997.

[GP99] C. Gonzalez and J. Peters. Localized hierarchy surface splines. In S.N. Spencer J. Rossignac, editor, *ACM Symposium on Interactive 3D Graphics*, 1999.

[Gre74] J. A. Gregory. *Smooth interpolation without twist constraints*, pages 71–88. Academic Press, 1974.

[JLW05] Shuangshuang Jin, Robert R. Lewis, and David West. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21(1-2):71–82, 2005.

[KP07] K. Karčiauskas and J. Peters. Concentric tesselation maps and curvature continuous guided surfaces. *Computer Aided Geometric Design*, 24(2):99–111, Feb 2007.

[KPN1] K. Karčiauskas and J. Peters. Guided spline surfaces. *Computer Aided Geometric Design*, pages 1–20, 2009 N1.

[KPR04] K. Karciauskas, J. Peters, and U. Reif. Shape characterization of subdivision surfaces – case studies. *Computer-Aided Geometric Design*, 21(6):601–614, july 2004.

[Loo92] Charles Teorell Loop. *Generalized B-spline surfaces of arbitrary topological type*. PhD thesis, University of Washington, 1992.

[Loo04] C. Loop. Second order smoothness over extraordinary vertices. In *Symposium on Geometry Processing*, pages 169–178, 2004.

[LS08a] Charles Loop and Scott Schaefer. Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches. *ACM Trans. Graph.*, 27(1):1–11, 2008.

[LS08b] Charles T. Loop and Scott Schaefer. G^2 tensor product splines over extraordinary vertices. *Comput. Graph. Forum*, 27(5):1373–1382, 2008.

[MKP07] Ashish Myles, Kęstutis Karčiauskas, and Jörg Peters. Extending Catmull-Clark subdivision and PCCM with polar structures. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 313–320, Washington, DC, USA, 2007. IEEE Computer Society.

[MNP08] A. Myles, T. Ni, and J. Peters. Fast parallel construction of smooth surfaces from meshes with tri/quad/pent facets. In *Symposium on Geometry Processing, July 2 - 4, 2008, Copenhagen, Denmark*, pages 1–8. Blackwell, 2008.

[MP09] A. Myles and J. Peters. Bi-3 C^2 polar subdivision. *ACM Transactions on Graphics*, 2009.

[Myl08] Ashish Myles. *Curvature-Continuous Bicubic Subdivision Surfaces for Polar Configurations*. PhD thesis, University of Florida, December 2008.

[PBP02] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline techniques*. Mathematics and Visualization. Springer-Verlag, Berlin, 2002.

[Pet91] J. Peters. Smooth interpolation of a mesh of curves. *Constructive Approximation*, 7:221–247, 1991. Winner of SIAM Student Paper Competition 1989.

[Pet95] J. Peters. C^1-surface splines. *SIAM Journal on Numerical Analysis*, 32(2):645–666, 1995.

[Pet02] J. Peters. Geometric continuity. In *Handbook of Computer Aided Geometric Design*, pages 193–229. Elsevier, 2002.

[Pet04] J. Peters. Mid-structures of subdividable linear efficient function enclosures linking curved and linear geometry. In Miriam Lucian and Marian Neamtu, editors, *Proceedings of SIAM conference, Seattle, Nov 2003*. Nashboro, 2004.

[Pet08] J. Peters. PN-quads. Technical Report 2008-421, Dept CISE, University of Florida, 2008.

[PK09] Jörg Peters and K. Karčiauskas. An introduction to guided and polar surfacing. In *Mathematics of Curves and Surfaces*, pages 1–26, 2009. Seventh International Conference on Mathematical Methods for Curves and Surfaces Toensberg, Norway.

[PR98] J. Peters and U. Reif. The 42 equivalence classes of quadratic surfaces in affine n-space. *Computer-Aided Geometric Design*, 15:459–473, 1998.

[PR08] J. Peters and U. Reif. *Subdivision Surfaces*, volume 3 of *Geometry and Computing*. Springer-Verlag, New York, 2008.

[PW08] J. Peters and X. Wu. Net-to-surface distance of subdivision functions. *JAT*, page xx, 2008. in press.

[SAUK04] Le-Jeng Shiue, Pierre Alliez, Radu Ursu, and Lutz Kettner. A tutorial on cgal polyhedron for subdivision algorithms. In *2nd CGAL User Workshop*, 2004. http://www.cgal.org/Tutorials/Polyhedron/.

[SJP05] Le-Jeng Shiue, Ian Jones, and J. Peters. A realtime GPU subdivision kernel. In Marcus Gross, editor, *Siggraph 2005, Computer Graphics Proceedings*, Annual Conference Series, pages 1010–1015. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2005.

[Sta98] J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In M. Cohen, editor, *SIGGRAPH 98 Proceedings*, pages 395–404. Addison Wesley, 1998.

[SW07] S. Schaefer and J. Warren. Exact evaluation of non-polynomial subdivision schemes at rational parameter values. In *PG '07: 15th Pacific Conference on Computer Graphics and Applications*, pages 321–330, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[SZBN03] Thomas W. Sederberg, Jianmin Zheng, Almaz Bakenov, and Ahmad Nasri. T-splines and T-NURCCs. In Jessica Hodgins and John C. Hart, editors, *Proceedings of ACM SIGGRAPH 2003*, volume 22(3) of *ACM Transactions on Graphics*, pages 477–484. ACM Press, 2003.

[VPBM01] Alex Vlachos, Jörg Peters, Chas Boyd, and Jason Mitchell. Curved PN Triangles. In *I3D 2001: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 159–166, 2001.

[vW86] J. van Wijk. Bicubic patches for approximating non-rectangular control-point meshes. *Computer Aided Geometric Design*, 3(1):1–13, 1986.

[WP04] X. Wu and J. Peters. Interference detection for subdivision surfaces. *Computer Graphics Forum, Eurographics 2004*, 23(3):577–585, 2004.

[WP05] X. Wu and J. Peters. An accurate error measure for adaptive subdivision surfaces. In *Proceedings of The International Conference on Shape Modeling and Applications 2005*, pages 51–57, 2005.

[WW02] J. Warren and H. Weimer. *Subdivision Methods for Geometric design.* Morgan Kaufmann, New York, 2002.

[YNM$^+$] Young In Yeo, Tianyun Ni, Ashish Myles, Vineet Goel, and Jörg Peters. Parallel smoothing of quad meshes. *The Visual Computer*, pages x–x. accepted, in press, TVCJ-267.

[ZS00] D. Zorin and P. Schröder, editors. *Subdivision for Modeling and Animation*, Course Notes. ACM SIGGRAPH, 2000.